

PA 01-308  
reference 2

(19) 日本国特許庁 (J P)

(12) 公表特許公報 (A)

(11) 特許出願公表番号

特表平11-514116

(43) 公表日 平成11年(1999)11月30日

(51) Int.Cl.<sup>6</sup>  
G 0 6 F 11/14

識別記号  
3 1 0

F 1  
G 0 6 F 11/14

3 1 0 B

審査請求 有 予備審査請求 有 (全 40 頁)

(21) 出願番号 特願平9-523695  
(86) (22) 出願日 平成8年(1996)12月11日  
(85) 審判文提出日 平成10年(1998)6月11日  
(86) 国際出願番号 PCT/US 96/19706  
(87) 国際公開番号 WO 97/23826  
(87) 国際公開日 平成9年(1997)7月3日  
(31) 優先権主張番号 08/570, 585  
(32) 優先日 1995年12月11日  
(33) 優先権主張国 米国 (US)

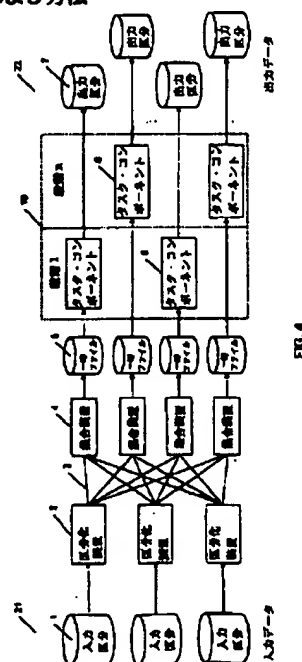
(71) 出願人 エービー イニティオ ソフトウェア コーポレーション  
アメリカ合衆国 01742 マサチューセッツ州, コンコード, パージニア ロード 477  
(72) 発明者 スタンフィル, クレグ  
アメリカ合衆国 02154 マサチューセッツ州, ワルサム, フローレンス ロード 62  
(74) 代理人 弁理士 平木 祐輔 (外2名)

最終頁に続く

(54) 【発明の名称】 コンポーネント・ベースの並列アプリケーションでチェックポイントを増加するための過剰区分化 (overpartitioning) システムおよび方法

(57) 【要約】

コンピュータ・プログラムによって実行される作業を、チェックポイントがより頻繁に行われるようにさらに小さな片に区分化するための2つの方法である。当初、並列タスクは $q$ 個の初期区分を持つ1つまたは複数のデータ・セットで開始し、入力データ・セットを、区分化要素のなんらかの組み合わせ(区分化装置/集合装置)によって $p$ 個の区分に分割し、データの $p$ 個の区分のそれぞれの上でコンポーネント・プログラムのインスタンスを実行し、1つまたは複数の出力ファイルのセットを作成し、各セットが区分化されたデータ・セットと見なされる。本発明は、新規の「過剰区分化」されたタスクを以下のように作成するためにこのようなタスクに適用される。(1) 区分化装置は、整数過剰区分化係数 $n$ に対して、その $q$ 個の入力を $n * p$ 個の区分に分割する「過剰区分化装置」と置換される。(2) コンポーネント・プログラムは、直列の $n$ 個の実行段階で実行され、コンポーネント・プログラムの $p$ 個のインスタンスは任意の時点で実行される。各段階では、コンポーネント・プログラムの各インスタンスが、入力データの1つの過剰区分



BEST AVAILABLE COPY

**【特許請求の範囲】**

1. (a) なんらかの整数過剰区分化係数  $n$  に対して、 $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割するステップと、

(b) コンポーネント・プログラムの各インスタンスが  $n * p$  個のデータ区分の内の 1 つを読み取り、出力データの対応する区分を作成する、直列の  $n$  個の実行段階でコンポーネント・プログラムの  $p$  個のインスタンスを実行するステップと、

(c)  $n$  個の実行段階のそれぞれの最後でチェックポイント設定を可能にするステップと、

を含む、並列処理システムで実行可能なコンピュータ・アプリケーションでチェックポイントの頻度を増加するための方法。

2.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割するステップが、さらに、

(a) データの  $n * p * q$  個のチャネルを出力するように構成される区分化プログラムを、 $q$  個のデータ入力に適用するステップと、

(b)  $n * p$  個のデータ区分を生成するために、区分化プログラムによって出力されるデータの  $n * p * q$  個のチャネルに集合プログラムを適用するステップと、

を含む、請求項 1 記載の方法。。

3.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割するステップが、さらに、

(a)  $q$  個のデータ入力を  $n$  個のデータ・セグメントに分割するために  $p$  個のカーソルを使用するステップと、

(b) 各データ・セグメントの最後を示すためにファイル終了信号を回帰的に生成するステップと、

を含む、請求項 1 記載の方法。。

4. 並列処理システム上で実行可能なコンピュータ・アプリケーションでのチェックポイントの頻度を増加するためのコンピュータ・プログラムであって、

(a) なんらかの整数過剰区分化係数  $n$  に対して、 $q$  個のデータ入力を  $n * p$

個のデータ区分に機能的に分割する機能と、

(b) コンポーネント・プログラムの各インスタンスが  $n * p$  個のデータ区分の内の1つを読み取り、出力データの対応する区分を作成する、直列の  $n$  個の実行段階でコンポーネント・プログラムの  $p$  個のインスタンスを実行する機能と、

(c)  $n$  個の実行段階のそれぞれの最後でチェックポイント設定を可能にする機能と、

を実行するためにコンピュータ・システムによって読み取られ、実行されると、コンピュータ・システムを構成する、コンピュータ・システムによって読取可能な媒体に記録されるコンピュータ・プログラム。

5.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割する機能が、さらに、

(a) データの  $n * p * q$  個のチャンネルを出力するように構成される区分化プログラムを、 $q$  個のデータ入力に適用する機能と、

(b)  $n * p$  個のデータ区分を生成するために、区分化プログラムによって出力されるデータの  $n * p * q$  個のチャンネルに集合プログラムを適用するステップと、

を含む、請求項4記載のコンピュータ・プログラム。

6.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割する機能が、さらに、

(a)  $q$  個のデータを  $n$  個のデータ・セグメントに分割するために  $p$  個のカーソルを使用する機能と、

(b) 各データ・セグメントの最後を示すためにファイル終了信号を回帰的に生成する機能と、

を含む、請求項4記載のコンピュータ・プログラム。

7. (a) なんらかの整数過剰区分化係数  $n$  に対して、 $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割する機能と、

(b) コンポーネント・プログラムの各インスタンスが  $n * p$  個のデータ区分の内の1つを読み取って、出力データの対応する区分を作成する、直列の  $n$  個の実行段階でコンポーネント・プログラムの  $p$  個のインスタンスを実行する機能と

(c)  $n$  個の実行段階のそれぞれの最後でチェックポイント設定を可能にする機能と、

を実行するために、記録媒体が特定の事前に定義された方法でコンピュータを動作させるように、並列処理システム上で実行可能なコンピュータ・アプリケーションでのチェックポイントの頻度を増加するためにコンピュータ・プログラムで構成されるコンピュータ読取可能な記録媒体。

8.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割する機能が、さらに、

(a) データの  $n * p * q$  個のチャネルを出力するように構成される区分化プログラムを、 $q$  個のデータ入力に適用する機能と、

(b)  $n * p$  個のデータ区分を生成するために、区分化プログラムによって出力されるデータの  $n * p * q$  個のチャネルに集合プログラムを適用する機能と、を含む、請求項7記載のコンピュータ読取可能な記録媒体。

9.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割する機能が、さらに、

(a)  $q$  個のデータ入力を  $n$  個のデータ・セグメントに分割するために  $p$  個のカーソルを使用する機能と、

(b) 各データ・セグメントを示すためにファイル終了信号を回帰的に生成する機能と、

を含む、請求項7記載のコンピュータ読取可能な記録媒体。

**【発明の詳細な説明】**

コンポーネント・ベースの並列アプリケーションでチェックポイントを増加するための過剰区分化 (overpartitioning) システムおよび方法

**発明の背景****技術分野**

本発明は、コンピュータ処理システムに関し、さらに特定するとコンピュータ・プログラムによって実行される作業を、チェックポイントがさらに頻繁に実行されるように、さらに小さな片に区分化するための方法に関する。

**関連技術の説明**

単一プロセッサ・コンピュータの計算速度は、過去 30 年間に渡って飛躍的に進歩してきた。しかしながら、多くの分野は、最高速度の単一プロセッサ・コンピュータさえも上回る計算速度を必要とする。例は、複数ユーザがコンピュータ・リソースに並行してアクセスし、応答時間はシステムが商業上受け容れられるためには遅くてはならないトランザクション処理である。別の例は、数百ギガバイトの情報が処理されなければならない、直列コンピュータ上でのデータ処理に数日または数週間を要する可能性のあるデータベース調査 (mining) である。したがって、このような問題を取り扱うために多岐に渡る「並列処理」システムが開発されてきた。この説明のために、並列処理システムは、ローカル (例えば、SMP コンピュータのようなマルチプロセッサ・システム) つまりローカルに分散された (例えば、クラスタまたは MPP として結合される複数プロセッサ)、または遠隔に、つまり遠隔に分散された (例えば、LAN または WAN ネットワークを介して結合される複数のプロセッサ)、またはその任意の組み合わせで複数の中央演算処理装置 (CPU) を使用するコンピュータ・システムの任意の構成を含む。

さらに多くのビジネス・アプリケーションは、データがある種の様式で変換す

る「コンポーネント・プログラム」のセットを含む。例えば、賃金台帳アプリケーションには、個人別就労時間集計用紙を支払小切手に変換したプログラムが含まれる可能性がある。このような構成要素を並列処理システム上での実行のため

のアプリケーションの一部として使用するためには、以下が共通した慣行である。

- 。
- (1) 並行の度合い  $p$  が選択される。大部分の場合、 $p$  はプロセッサの数と同一である。
- (2) コンポーネント・プログラムに対する1つまたは複数の入力、 $p$  個の入力ファイルのセットの間で分割される(「区分」)。
- (3) コンポーネント・プログラムの  $p$  個のインスタンスは、さまざまなプロセッサで同時に実行される。
- (4) コンポーネント・プログラムの各インスタンスは、入力データの1つの区分を処理する。

コンポーネント・プログラムが実行を終了すると、その出力のそれぞれは、 $p$  個の区分に分割される。その後、出力は、さらなるコンポーネント・プログラムのために入力として使用されるか、あるいは記憶装置に書き込まれる。

これらの区分内でのデータの編成は、大部分の場合、かなり重要である。つまり、多くの場合、データが適切に区分化されていないと、コンポーネント・プログラムは間違った結果を出す。さらに、各区分内のデータの順序がかなり重要であるのも、大部分の場合に当てはまる。例えば、多くのプログラムは、その入力(複数の場合がある)が何らかの様式でソートされることを必要とする。

データの正しい区分化を達成するためには、入力ファイルを取り、それをなんらかの規則に則って  $p$  個の部分に分割する別個の「区分化装置(partitioner)」プログラムを使用するのが一般的な慣行である。区分化の方法がコンポーネント・プログラムによって出される結果に影響を及ぼさない場合では、区分化装置はデータを任意の様式で分割することができる。区分化が、各レコードをレコード内のデータ値に基づいて  $p$  個の区分の内の1つに割り当てる数学的な関数に基づいているのも一般的である。

一般的なケースでは、入力のそれぞれは、必ずしもコンポーネント・プログラムのニーズに一致した方法ではないが、 $q$  個の初期区分に分割される。このよう

なデータを区分化するためには、一般的な慣行とは、区分化装置プログラムの  $q$

個のコピーを実行することであり、各コピーは $q$ 個の初期区分の1つを入力として与えられる。区分化プログラムの $q$ 個のコピーのそれぞれは $p$ 個の出力区分を作成するので、これは入力データを $p * q$ 区分に分割する。この問題を第2の「集合装置 (gatherer)」プログラムを利用して、 $q$ 個の区分 (区分化装置の各インスタンスに1つ) のグループを結合し、最終的な出力を出すことによって解決するのが一般的な慣行である。

アプリケーションは、いったん書き込まれると、任意の並列の度合い $p$ で実行可能となるのが望ましい。これは、時間が経過するに従って、さらに多数のプロセッサを備えるコンピュータの取得を必要とするデータの量が増加するために必要である。これを支持して、区分化装置／集合装置プログラムを、それらが任意の希望の数の区分 $p$ を作成できるように、およびコンポーネント・プログラムが正しく動作するように、パラメータ化することが一般的な慣行である。この特性が「スケーラビリティ」と呼ばれる。

図1は、 $q$ 個の初期区分を $p$ 個の出力区分に分割するための従来の技術の区分化装置／集合装置の図である。図1は、単一入力から読み取り、単一出力に書き込むコンポーネント・プログラムの場合のこのようなシステムを示す。入力データ21は、 $q = 3$ 初期区分1に区分化されている。希望の並列の度合いは $p = 2$ である。区分化装置プログラム2の3つのインスタンスが実行され、通信チャンネル3に沿って伝送される6つの中間結果を出す。これらの中間結果は、集合装置プログラム4の2つのインスタンスによって読み取られ、2つの一時ファイル5を作成する。それから、コンポーネント・プログラム6の2つのインスタンスが実行され、その内のそれぞれが別個の一時ファイル5から読み取り、別個の出力ファイル7に書き込む。出力ファイル7は出力データ・セット22の区分を構成する。もし複数の出力データ・セット21があれば、区分化装置2、通信チャンネル3、集合装置5、および一時ファイル6は入力データ・セットごとに一度複製される (区分化装置2の数を変えながら)。複数の出力データ・セット22を持つことも可能である。最後に、いくつかの個別入力ファイルが区分化されるのではなく、コンポーネント・プログラム (図示されていない) のすべてのインスタ

ンスに一斉送信されることが考えられる。

このアーキテクチャには多数の変形がある。例えば、一時ファイル5が対応する通信チャネルによって置き換えられるか、あるいは通信チャネル3が対応するファイルで置き換えられる可能性がある。

図2は、「事前区分化済み (pre-partitioned) データ」の重要な特別ケースを示す従来の技術のシステムの図である。このケースは、入力区分数  $q$  がコンポーネント・プログラムの希望の並列度合い  $p$  に等しく、入力データ 21 が区分化される方法がコンポーネント・プログラム6に適當である場合に発生する。このケースでは、データを区分化する必要はなく、区分化装置2、通信チャネル3、集合装置4、および一時ファイル5を排除することができる。区分化装置／集合装置要素2-5の排除により、かなりの計算上の作業とかなりの記憶スペースが節約されるため、この特別ケースを利用することは有利である。

コンポーネント・プログラム6が複数の入力を持つ場合、その入力の内のいくつかは、図1に図示される区分化を必要とし、それ以外は、図2のように「事前区分化」されることが考えられる。

複数のコンポーネント・プログラムを備える複雑なアプリケーションを構築するために、人は、図1または図2に示されるような構造を持つ複数の計算単位（「タスク」）をリンクさせる。このリンクは、処理のある段階の出力区分7を、次の段階の対応する入力区分1に割り当てることによって実行される。

複雑なアプリケーションの場合は、システム故障が発生しても、アプリケーション全体を始めから実行し直すのではなく、アプリケーションが計算の中間状態を捕捉する「チェックポイント」から続行できるようにする「チェックポイント」機構を備えることが望ましい。チェックポイントに関する一般的な説明、および本発明が貴重となる可能性がある改善型チェックポイント・システムの具体的な例については、本発明の譲受け人に譲渡された「計算の状態を再構築するための方法およびシステム (Methods and Systems for Reconstructing the State of a Computation)」と題する同時係属の特許明細書を参照する。

チェックポイントの制約の内の1つとは、システムが「静止」している場合、つまりアプリケーション・プログラムが実行していない場合にだけチェックポイ



ントが作成されるという点である。システムは、タスクの間で静止しており、したがって各タスクの端でチェックポイントを作成できるように配置できる。また、集合装置プログラム 4 が実行を終了した直後に、タスクにとって内部に 1 つのチェックポイントを作成することも可能である。ただし、これらの技法がチェックポイントに十分に頻繁な機会を生み出さない可能性もあると考えられる。例えば、タスクのコンポーネント・プログラム 6 が 10 時間実行すると、アプリケーションは、どのチェックポイントも可能ではないときに必然的に 10 時間という期間を持つだろう。その 10 時間の間の任意の時点で故障が発生すると、アプリケーション全体を最初から起動し直さなければならない。

したがって、並列構成要素ベースのアプリケーション内でさらに多くの頻繁なチェックポイントを許す方法に対するニーズがある。本発明は、並列処理システムで実行中のアプリケーションに特に有効であり、分散処理システム上で実行するアプリケーションにも有効であるこのような方法を提供する。

#### 発明の要約

本発明は、複雑な並列アプリケーション内で、それがさらに頻繁に静止（したがって、チェックポイント設定可能（checkpointable））となるように、タスクを修正するための 2 通りの方法を含む。

当初、並列タスクは  $q$  個の初期区分を持つ 1 つまたは複数の入力データ・セットで開始し、入力データ・セットを、区分化要素（つまり、区分化装置／集合装置）のなんらかの組み合わせによって  $p$  個の区分に分割し、データの  $p$  個の区分のそれぞれでコンポーネント・プログラムのインスタンスを実行し、1 つまたは複数の出力ファイルを作成し、そのそれぞれのセットが区分化されたデータ・セットと見なされる。本発明は、新規の「過剰区分化」されたタスクを以下のように作成するためにこのようなタスクに適用される。

(1) 区分化装置は、整数過剰区分化係数  $n$  のためにその  $q$  個の入力を  $n * p$  個の区分に分割する「過剰区分化装置」と置換される。

(2) コンポーネント・プログラムは、 $n$  個の実行段階が直列に実行され、コンポーネント・プログラムの  $p$  個のインスタンスは任意の時点で実行される。各段

階では、コンポーネント・プログラムの各インスタンスが、入力データの1つの過剰区分(overpartition)を読み取り、出力データの1つの区分を作成する。

(3)  $n$ 個の実行段階のそれぞれの最後で、システムは静止しており、チェックポイント設定をすることができる。

本発明は2つの好ましい実施例を含む。1つの実施例とは、過剰区分化された中間ファイルを作り出すために、既知の区分化装置プログラム、通信チャネル、および集合装置プログラムを使用して入力データを「明示的に過剰区分化」することである。この実施例は、図1に図示される形式のタスクに適用可能である。第2実施例は、オリジナル入力データの連続サブセットを連続的に読み取るためにコンポーネント・プログラムを手配することによって入力データを「動的に過剰区分化する」ことである。この実施例は、図2に図示される形式の多くのタスクに適用可能である。

本発明の好ましい実施例の詳細は、添付図面および以下の説明に述べられている。いったん本発明の詳細が既知になると、数多くの補助的なイノベーションおよび変化が当業者に明らかになるだろう。

#### 図面の簡単な説明

図1は、 $q$ 個の初期区分を $p$ 個の出力区分に分割するための従来の技術による区分化装置／集合装置の図である。

図2は、「事前区分化データ」の特別ケースを示す従来の技術のシステムの図である。

図3aは、本発明に従った過剰区分化システムの適用の前のオリジナル・タスクの全体的な図である。

図3bは、本発明に従った過剰区分化システムの適用後の図3aのタスクの全体的な図である。

図4は、本発明に従った明示過剰区分化システムの好ましい実施例を示す、図3bの過剰区分化システムのより詳細な図である。

図5aおよび図5bは、データ・フロー、および本発明に従った明示過剰区分化システムを実現するためのシステムのプロセッサ、通信、および記憶域の関係

性を示すさらに詳細な図である。

図6は、本発明に従った動的過剰区分化システムの好ましい実施例を示す、図3bの過剰区分化システムのさらに詳細な図である。

図7は、データ・フロー、および本発明に従った動的過剰区分化システムを実現するためのシステムのプロセッサ、通信および記憶域の関係性を示す、さらに詳細な図である。

図8は、本発明に従った動的過剰区分化システムでカーソルおよびモニタを実現するためのシステムのデータ・フロー、機能、および通信の関係性を示す図である。

さまざまな図の類似した番号および名称は、類似した要素を示す。

#### 本発明の詳細な説明

本説明を通して、好ましい実施例および図示される例は、本発明に対する制限としてより、むしろ例と見なされる必要がある。

#### 概要

図3aは、本発明に従った過剰区分化システムの適用の前のオリジナル・タスクの全体的な図である。図3aは、 $q$ 個の入力区分1を備える1つまたは複数の入力データ・セット21で開始し、それらを区分化要素8のなんらかの組み合わせ（つまり、区分化装置／集合装置）によって $p$ 個の区分12に分割し、データの $p$ 個の区分12のそれぞれでコンポーネント・プログラム6のインスタンスを実行し、出力データ22の1つまたは複数の出力区分7を作成する並列タスク11を示す。

図3bは、本発明に従った過剰区分化システムの適用後の、図3aのタスクの全体的な図である。新規の過剰区分化されたタスク13は、以下のように作成される。

(1) 区分化装置8は、なんらかの整数過剰区分化係数 $n$ のために、その $q$ 個の入力区分をデータの $n * p$ 個の区分12に分割する「過剰区分化装置」9と置換される。

(2) コンポーネント・プログラム6は、 $n$ 個の実行段階10が直列に実行され

、コンポーネント・プログラムの  $p$  個のインスタンスは任意の時点で実行される。各段階で、コンポーネント・プログラム 6 の各インスタンスは、入力データ 2 1 の 1 つの区分 1 2 を読み取り、出力データ 2 2 の 1 つの区分 7 を作成する。入力データ・ファイル 2 1 は、順次、階層、関係などの好ましいフォーマットとすることができる。

(3)  $n$  個の実行段階のそれぞれの最後では、システムは静止しており、チェックポイント設定をすることができる。

ここでの重要な特徴とは、大部分の場合には、コンポーネント・プログラム 6 を実行するために必要とされる時間が処理されるデータの量に強く結び付けられている、例えば、データの半分は約半分の時間で処理することができるという点である。したがって、データの量の  $n$  倍の削減は、対応する係数  $n$  分、コンポーネント・プログラム 6 のインスタンスの実行時間を縮小し、結果的に係数  $n$  分、チェックポイント設定機会の間の間隔を削減する可能性がある。

本発明は、2 つの好ましい実施例を含む。1 つの実施例は、過剰区分化された中間ファイル 5 を作成するために、既知の区分化プログラム 2、通信チャネル 3、および集合装置プログラム 4 を使用して入力データ 2 1 を「明示的に過剰区分化」することである。この実施例は、図 1 に図示される形式のタスクに適用可能である。第 2 実施例は、オリジナルの入力データ 2 1 の連続サブセットを連続的に読み取るために、コンポーネント・プログラム 6 を手配することによって入力データ 2 1 を「動的に過剰区分化する」ことである。この実施例は、図 2 に図示される形式の多くのタスクに適用可能である。

#### 明示過剰区分化

明示過剰区分化の方法は、中間「過剰区分化された」ファイル 5 のセットを作り出すために既存の区分化装置 2、通信チャネル 3、および集合装置 4 を使用してから、過剰区分化されたファイルと対照して複数の実行段階でコンポーネント・プログラム 6 を実行する。この方法は、以下の要件が満たされる場合に使用される。

(1) 方法は、入力データ・セット 2 1、並列の希望の度合い  $p$ 、および過剰区

分化の希望の度合い  $n$  から  $q$  個の入力区分 1 を提示されなければならない。

(2) 区分化装置プログラム  $p$  が、少なくとも  $n * p$  個の出力区分を作り出すために有効に構成されることが必要である。

(3) コンポーネント・プログラム 6 が「スケーラブル」であることが必要である。つまり、アプリケーションが並列の任意の度合い  $p$  に対して有効な結果を出すことが理解されなければならない。

図 4 は、図 3 b の過剰区分化システムのさらに詳細な図であり、本発明に従った明示過剰区分化システムの好ましい実施例である。前記事実が真実の場合、明示過剰区分化は、以下のように実現できる。

(1) 区分化装置 2 の  $q$  個のインスタンスが、 $q$  個の入力区分 1 と作用するように設定される。区分化装置 2 の各インスタンスは、 $n * p$  個の区分を作成するために構成される。

(2) 集合装置 4 の  $n * p$  個のインスタンスは、区分化装置 2 の出力と作用するように設定される。

(3) 区分化装置 2 の  $q$  個のインスタンスは、図示されるように、 $n * p * q$  個の通信チャネル 3 を使用して  $n * p$  個のインスタンスと接続される。

(4) すべての区分化装置 2 および集合装置 4 が実行され、 $n * p$  個の一時ファイル 5 を作成する。

(5)  $n$  個の実行段階 10 は、前記のように実行される。各段階では、コンポーネント・プログラム 6 の  $p$  個のインスタンスが実行され、それぞれのこのようなインスタンスは、従来の技術におけるように、一時ファイル 5 の内の 1 つからデータを消費し、データを出力区分ファイル 7 の内の 1 つに書き込む。出力は、 $n * p$  個のファイルを構成する。

多くのケースでは、構成要素プログラム 6 の論理は、 $n$  個の連続実行段階 10 によって作り出される出力が、連結される場合には、 $n$  の値に関係なく、非過剰区分化されたケースでのように ( $n * p$  よりむしろ)  $p$  個の出力ファイルの同じセットを作り出すようなものがある。このケースでは、 $p$  個の出力ファイルのセットは、コンポーネント・プログラム 6 の各インスタンスによってその連続実行段

階10の間に出される出力を追加するだけで作成することができる。これには、管理されなければならない出力ファイルの数を削減するという利点がある。

従来の技術でのように、このアーキテクチャには数多くの変形がある。例えば、一時ファイル5が対応する通信チャネルによって置換されたり、通信チャネル3が対応するファイルと置換することができる。

前記ステップは、手動によって（例えば、アプリケーションの全体的な実行を制御するために使用されるソフトウェアを書き直すことによって）または自動的に実現できる。好ましい実施例では、自動ケースは、タスクごとに以下の情報（または同等な情報）がコンピュータで読取り可能な形式で指定されることを必要とする。

（1）並列の希望する度合い $p$ 、および過剰区分化の希望する度合い $n$

（2）コンポーネント・プログラム6のアイデンティティ

（3）コンポーネント・プログラム6への入力データ21を有するファイルのアイデンティティ

（4）タスクの入力データ・セット21ごとに、以下の追加情報：

1）その入力区分1を有するファイルのアイデンティティ

2）その入力に適切な区分化装置2のアイデンティティ

3）その入力に適切な集合装置4のアイデンティティ

4） $p$ 個の一時ファイル5を記憶することができる適当な記憶装置のアイデンティティ

（5）出力データ・セット22ごとに、 $p$ 個の出力区分ファイル7を記憶することができる適当な記憶装置のアイデンティティ

（6）区分化装置2を実行することができる $q$ 個の「区分」プロセッサの識別子

（7）集合装置4を実行することができる $p$ 個の「集合」プロセッサの識別子（これらは、希望する場合、区分プロセッサと同じとすることができる）。

（8）コンポーネント・プログラム6を実行することができる $p$ 個の「ワーク」プロセッサの識別子（これらは、希望する場合、区分プロセッサまたは集合プロセッサと同じとすることができる）。

（9）区分化装置2、集合装置4、およびコンポーネント・プログラム6の場合

必要に応じてそれらを実行させ、それらをデータ・ファイルおよび通信チャネルとして接続させるためのこのような情報、および区分化装置2のケースでは、すべて既知の様式で任意の数の区分を作り出すためにそれらをどのようにして構成するのかに関する情報。好ましい実施例では、この情報は、接続先のデータ・ファイル／サブルーチンのアイデンティティ（加えて、区分化装置2の場合、区分の数）が指定され、希望すればプログラムを実行する「引数ライン・パラメータ」のシーケンスを戻すサブルーチンとして表される。

いったん上記情報（またはその均等物）が提供されると、明示過剰区分化アルゴリズムを実行することができる。図5 aおよび図5 bは、データ・フロー、および本発明に従って明示過剰化区分化システムを実現するためのシステムのプロセッサ、通信、および記憶域の関係性を示す詳細な図である。

図5 aを参照すると、タスクのために入力データ21を有するファイルが、好ましい実施例で次に示すように反復的に処理される。

- (1) 手元にあるデータ入力ファイル21の入力区分1の数 $q$ を識別すること。
- (2)  $n * p * q$ 個の通信チャネル3を下記によって作成すること、
  - 1) 第1区分プロセッサ23で第1の $n * p$ 個のチャネルを起動し、次の区分プロセッサ23などで次の $n * p$ 個のチャネルを起動すること、
  - 2) サイクルが第1集合プロセッサ24で開始し直して繰り返す $n * p$ 個の通信チャネルが割り当てられるまで、第1の $n$ 個のチャネル3を第1集合プロセッサ24に接続し、次の $n$ 個のチャネルを次の集合プロセッサ24などに接続すること、
- (3) 区分化装置2の $q$ 個のインスタンスを実行すること、ただし、
  - 1) 各インスタンスが区分プロセッサ23の1つに割り当てられる。
  - 2) 各インスタンスが、 $n * p$ 個の出力区分を作成するために構成される。
  - 3) 各インスタンスが、そのプロセッサ内で起動する通信チャネル3のすべてに接続する。
- 4)  $i$ 番目のインスタンスが、入力データ・セット21の $i$ 番目の入力区分1が

ら読み取る。

(4) 集合装置4の $n * p$ 個のインスタンスを実行すること、ただし、

1)  $n$ 個のインスタンスが集合プロセッサ24のそれぞれに割り当てられる。各プロセッサ上の $i$ 番目のインスタンスが、未接続の通信チャネル3がなくなるまで、 $i$ 番目の入信通信チャネル3、 $n + i$ 番目のチャネル3、 $2n + i$ 番目のチャネル3などに接続する。

2)  $k$ 番目の集合プロセッサ24上で実行するインスタンスが、「一時ファイル」記憶装置25のセット内の記憶装置に $k$ 番目の一時ファイル5を書き込む（さらに高い性能のために、 $k$ 番目の集合プロセッサ24で実行する各インスタンスは、別個の $k$ 番目の記憶装置上の $k$ 番目の一時ファイル5に書き込む）。

(5) 実行を終了するために区分化装置2と集合装置4のすべてのインスタンスを待機してから、通信チャネル3を削除すること、

(6) 入力データ・ファイル21の次のセットに継続する前に、オプションでシステムにチェックポイントを設定すること（この時点で、手元のデータ入力ファイル21の過剰区分化は完了している。チェックポイントの設定が、必ずしも、各チェックポイント設定可能なイベントで実際に実行される必要はないことに注意する）。

(7) 入力データ・ファイル21の任意の次のセットのために繰り返されるステップ(1) - (6)。

図5bを参照すると、 $n$ 個の実行段階10は、好ましい実施例で次に示すように一時ファイル5内のデータに対して実行される。

(1) コンポーネント・プログラム6の $p$ 個のコピーを、 $n$ 個の実行段階10のそれぞれの間にワーク・プロセッサ26の内のそれぞれで1つ実行すること。

(2)  $i$ 番目の実行段階10の間に、 $j$ 番目のワーク・プロセッサ26上で実行される、コンポーネント・プログラム6のインスタンスが、一時記憶装置25から $i$ 番目の一時ファイル5を読み取り、出力記憶装置26上の $i$ 番目の出力区分ファイル7に書き込む（さらに高い性能のために、 $j$ 番目のワーク・プロセッサ26上で実行する各インスタンスは、 $j$ 番目の一時記憶装置から $i$ 番目の一時フ



ファイル5を読み取り、別個のj番目の出力記憶装置27上のi番目の出力区分ファイル7に書き込む)。

(3) オプションで、一時ファイル5をそれらが消費される実行段階10内で削除すること。

(4) システムにチェックポイントが設定される、n個の実行段階10のそれぞれで実行するコンポーネント・プログラム6のすべてのp個のインスタンスの終了を待機すること。

このようにして、n個の実行段階10のそれぞれの最後で、コンポーネント・プログラム6のp個のインスタンスは、合計入力データ21のn分の1を処理してから、静止し、チェックポイントを設定することができる。例えば、 $q=2$ および $p=5$ の場合、従来の技術の下では、各コンポーネント・プログラム6は、1サイクル時間内で合計入力データ21の5分の1を処理するだろう。つまり、すべてのコンポーネント・プログラム6の並行処理を想定し、データのすべてが同じサイクル時間内に処理される。ただし、チェックポイント設定は、サイクルが終了するまで発生しない。本発明を使用し、3という過剰区分化係数を選択することにより、処理対象の区分の総数は $3 * 5 = 15$ となる。ただし、それぞれが従来の技術の区分の約3分の1のサイズの5個の区分だけが各実行サイクルで処理される。線形処理時間を想定し、各コンポーネント・プログラム6のサイクル時間は、従来の技術のサイクル時間の約3分の1となる。チェックポイント設定が、このように縮小された各サイクルの後に実行されてから、次の実行段階10が開始される。チェックポイント設定段階および実行段階10は、それからすべての過剰区分が処理されるまで繰り返される。

#### 動的過剰区分化

前記のように、いくつかのケースでは、入力データ21は、コンポーネント・プログラム6に協和的な方法で初期に区分化され、そのケースでは図2に図示される最適化が可能になる(つまり、区分化装置2、通信チャネル3、集合装置4、および一時ファイル5が排除される)。明示過剰区分化方法は、必然的に、区分化装置などの使用を必要とするため、事前区分化された最適化の利点を失う。

本発明の第2実施例は、一時的な過剰区分化されたファイル5を作成するために区分化装置などに頼らない。代わりに、第2実施例は、効果的に適所の入力データ・ファイル1を区分化し直す。この方法が「動的過剰区分化」と呼ばれる。

図6は、図3bの過剰区分化システムのさらに詳細な図であり、本発明に従った動的過剰区分化システムの好ましい実施例を示す。この方法は、コンポーネント・プログラム6が受け容れることができる区分が、入力ファイル1をデータ・セグメントと呼ばれる順次区分27に分割することによって入手される場合に使用される（このようなセグメントを作成する方法は、以下に説明される）。

動的過剰区分化動作28は、コンポーネント・プログラム6が入力区分1のデータ・セグメント27にアクセスできるようにするという効果を持ち、各入力区分1からの1つのセグメントが、明示過剰区分化に類似した方法で、複数の実行段階10のそれぞれのコンポーネント・プログラム6によって処理される。動的過剰区分化動作28は、一時ファイルを作成しないでこれを実行する。動的過剰区分化の本質とは、コンポーネント・プログラム6の各インスタンスを、入力データ・ファイル1のデータ・セグメント27を読み取らせるが、このようなインスタンスがそれが完全なデータ・セット1を読み取っていると考えることができるように「騙す」ことである。

図7は、データ・フロー、および本発明に従った動的過剰区分化を実現するためのシステムのプロセッサ、通信、および記憶域の関係性を示すさらに詳細な図である。動的過剰区分化は、次に示すように、好ましい実施例において実現される。

(1) 入力区分1ごとに固定「カーソル」29およびモニタ30を作成すること（図6の動的過剰区分化動作28は、カーソル29およびモニタ30を有する。カーソル29は、入力区分1内の位置（例えば、バイト・オフセット）を記憶する。これによって、コンポーネント・プログラム6は、カーソル29によって現在指されている位置で入力ファイル1からデータを読み取ることができる。モニタ30は、カーソル29によって指されるデータ・レコードを検査し、適当なときにカーソル29に、入力区分1内の次のデータ・レコードを返すのではなく「

ファイル終了」(EOF)を示すように命令することができる。)

(2) 各実行段階10のコンポーネント・プログラム6のp個のインスタンスを実行すること。

(3) (コンポーネント・プログラム6が、入力区分1に直接にアクセスするこ

とができるようにするのではなく)カーソル29を介するだけでデータ・セグメント27のデータにアクセスすること。

(4) モニタ30がデータ・セグメント27の終わりを検出すると、以下にさらに説明するようにEOFを示すようにカーソル29に指示すること(カーソル29は実際には入力区分1の最後にはないため、このイベントは「人工的なファイル終了」と呼ばれる)。

(5) (そのそれぞれが、対応するデータ・セグメント27内のデータのすべてを処理した)コンポーネント・プログラム6を終了すること。

(6) 好ましくはチェックポイント設定システムの一部を構成する2段階コミット・プロトコルを使用して、それ以降、カーソル29およびモニタ30の内容を固定記憶域に書き込むこと。

(7) オプションで、次のデータ・セグメント27に進む前にシステムにチェックポイントを設定すること。

(8) 次の実行段階10の開始時に、カーソル29およびモニタ30の内容をワーク記憶域に読み取ること。

(9) データにアクセスするために対応するカーソル29を使用してコンポーネント・プログラム6のさらなる実行を開始すること(各カーソル29は、ここでは入力区分1の中間部分、さらに具体的にはその入力区分1のデータ・セグメント27の次のセグメントの開始を参照する。したがって、処理はまさにそれが前の実行段階10で終了したところで起動する。各実行段階10の出力は、各事前実行段階出力に追加され、休止なしに最初から最後まで入力データ・ファイル21を通してアプリケーションが進んだように見えるようにする。)

(10) すべてのカーソル29がその入力区分1の「本当の」最後に到達したときにすべての実行を終了する(つまり、すべてのデータが処理されればそれ以上

の段階が実行されない)。

モニタ30がカーソル29に人工的なEOFを信号で知らせるように指示するときの決定は、コンポーネント・プログラム6によって利用される区分化機構に依存する。複数のケースがある。

最初に、コンポーネント・プログラム6の入力データが任意の様式で区分化さ

れることが考えられる。これは、例えば、あるレコードの処理が完全に次のレコードの処理から独立している場合に発生することがある。このケースでは、セグメント境界は任意の便宜的なレコード境界に該当する。

第2に、コンポーネント・プログラム6は、なんらかのキー（例えば、従業員番号）の同じ値が設定されるすべてのレコードが同じセグメント27に割り当てられることを要求する場合が考えられる。このケースでは、入力データ1は、そのキーに応じてソートされる必要がある。このケースでは、セグメント境界は、同一のキー値が設定される連続レコードの実行の間の境界に該当するように制約されるのが好ましい。

第3に、コンポーネント・プログラム6が複数の入力から読み取ることが考えられ、指定されたキー値が設定されるすべてのレコードが同じ実行段階で処理されることが必要とされる。このケースでは、すべてのこのような入力ファイル中のデータは、入力キーでソートされる必要があり、コンポーネント・プログラム6に先行するさまざまなモニタ30は、全てのカーソル29が、同じキー値境界上でそのデータを区分するように、人工的なEOF信号を調整しなければならない。

最後に、モニタ30が人工的なEOFを導入するために適当な場所を探し始めるときを判断するためにはなんらかの機構が必要とされる。動的過剰区分化の場合では、過剰区分化の度合いが事前に決定される必要がないことに注意する。コンポーネント・プログラム6は、いくつかのイベントが発生するまで実行される（例えば、特定のデータ値が入力データ・ストリーム内で発生するか、指定された時間間隔が期限切れになるか、指定された数のバイトまたはレコードが処理されるか、あるいは入力データ・ファイル・サイズの指定されたパーセンテージが

処理されるかなど)。

これらの要件は、本発明に従った動的過剰区分化システム内でカーソル29およびモニタ30を実現するためのシステムのデータ・フロー、機能、および通信の関係性を示す図である図8に図示されるアーキテクチャによって満たされる。

(1) すべてのモニタ30は、相互通信のためにグローバルな「トリガ・チャンネル」31を共用する。

(2) 特定のキー値で「調整」しなければならないすべてのモニタ30は、調整チャンネル32を介して接続される(複数の調整チャンネル32がある場合がある)。

(3) 任意のモニタ30は、任意の時点で、トリガ・チャンネル31上で「位相破損」メッセージを一斉通信する。メッセージは、トリガ・チャンネル31上ですべての他のモニタ30によって受け取られる。典型的には、このようなイベントは、ある一定期間の時間が経過した後に、一定のデータ量がモニタを通り抜けた後に、あるいは何らかのキー値がレコード内で見られた後に起こる。

(4) いったん位相破損メッセージが受信されると、各モニタ30は、そのカレント・データ・セグメント27から実際に読み取られた最後のレコードからキーを入手し、その調整チャンネル32上でそのキーを指定される「マスタ・モニタ」に伝送する。

(5) 各マスタ・モニタは、1つのキーが(それ自体を含む)その調整チャンネル32上で各モニタ30から受け取られるまで待機してから、このセットからの最大キー値を検出する。このキーが「ブレイクポイント・キー」と呼ばれる。

(6) 各マスタ・モニタは、その調整チャンネル32上ですべてのモニタ30にブレイクポイント・キーを一斉通信する。

(7) 各モニタ30は、それからその対応するカーソル29が、ブレイクポイント・キーの値を超えるキーに到達するまで対応するコンポーネント・プログラム6にデータを送達し続ける。このポイントでは、人工的なEOF信号がそのモニタ30によって作成され、現在の実行段階10の最後につながる。

(8) (コンポーネント・プログラム6が単一データ入力ファイル21から読み

取る場合などの) モニタを調整する必要がない特別なケースでは、各調整チャンネル32が単一モニタ30を含む(つまり、モニタ間での情報の共用は必要とされない)。上記アルゴリズムは、それ以外の場合変更されない。代替形式では、調整チャンネル32はまったく設定されない。

(9) (経過した時間、またはレコードの数などに基づくような) キーに関係なく位相破損が作成される特別なケースでは、モニタ30は、トリガ・チャンネル31上で位相破損メッセージの受信直後に人工的なEOFを作成するだけである。

カーソル29およびモニタ30の組み合わせは、少なくとも以下の3通りの内の1つで、コンポーネント・プログラム6と入力区分1の間におくことができる。

(1) コンポーネント・プログラム6は、カーソル29およびモニタ30を実現するサブルーチン・ライブラリを使用して書き込まれるか、あるいは再度書き込まれる。これには、コンポーネント・プログラム6用のソース・コードへのアクセスが必要とされ、時間を消費する可能性があるが、最高の性能を達成する可能性がある。

(2) コンポーネント・プログラム6は、既知の様式で基礎を成す入出力機構への呼出しに割り込む実行時ライブラリに対照してリンクされる。例えば、「開く」ための呼出しが割り込まれ、「カーソルを開く」ための呼出しに変換される。「読み取る」ための呼出しが妨害され、「カーソルから読み取る」ための呼出しに変換される。これには、コンポーネント・プログラム6のオブジェクト・コードへのアクセスが必要になるが、かなりの効率が見込まれる。

(3) 「アダプタ・プロセス」が各コンポーネント・プログラム6とその入力区分1の間におかれる。アダプタ・プロセスは入力区分1ファイルをカーソル/モニタ機構を介して読み取ってから、データを通信チャンネルまたは一時ファイルを介して既知の様式でコンポーネント・プログラム6に転送する。このオプションは、構成要素プログラム6の実行可能な(バイナリ)形式へのアクセスを必要とするだけであるが、入力区分1ファイルから通信チャンネルまたは一時ファイルにデータをコピーする費用のためにやや非効率的となる可能性がある。

要約すると、コンポーネント・プログラム6の各インスタンスを、一時に入力データ・ファイル1の1つのデータ・セグメント27だけを読み取るように「騙し」てから、各データ・セグメント27の最後で人工的なE O Fを作成することによってコンポーネント・プログラム6の「自然」終了を作成することにより、コンポーネント・プログラム6の終了を強制することによって、アプリケーションはさらに頻繁に静止することになり、このようにして頻繁なチェックポイント設定を本質的にサポートしないアプリケーションのさらに優れたチェックポイント設定を可能にする。

本発明は、このようにしてそれがさらに頻繁に静止（したがってチェックポイント設定可能）するように複雑な並列アプリケーションでタスクを修正するための2通りの方法を含む。本発明は、並列処理システム上で実行するアプリケーションに特に有効であり、分散処理システムで実行しているアプリケーションにも有効であるこのような方法を提供する。

本発明は、ハードウェアまたはソフトウェア、あるいは両方の組み合わせで実現することができる。ただし、好ましくは、本発明は、それぞれがプロセッサ、（揮発性メモリと不揮発性メモリ、または記憶装置要素、あるいはそのすべてを含む）データ記憶装置システム、少なくとも1つの入力装置、および少なくとも1つの出力装置を含む、プログラマブル・コンピュータ上で実行するコンピュータ・プログラムで実現される。プログラム・コードは、本明細書中に記述される機能を実行し、出力情報を生成するための入力データに適用される。出力情報は、既知の様式で1つまたは複数の出力装置に適用される。

各プログラムは、コンピュータ・システムと通信するために高水準手順プログラミング言語またはオブジェクト指向型プログラミング言語で実現されるのが好ましい。ただし、プログラムは、希望する場合、アセンブリ言語または機械語で実現することができる。いずれにせよ、言語はコンパイルされた言語または翻訳された言語とすることができる。

このような各コンピュータ・プログラムは、記録媒体または装置が、本明細書中に説明される手順を実行するためにコンピュータによって読み取られるときに

、コンピュータを構成し、動作するための汎用または特殊目的プログラマブル・コンピュータによって読取可能な（ROMまたは磁気ディスクのような）記録媒体または記録装置上に記録されるのが好ましい。発明的なシステムは、そのように構成される記録媒体によってコンピュータが本明細書中に説明される機能を特殊な事前に定義された方法で動作させる、コンピュータ・プログラムで構成されるコンピュータ読取可能な記録媒体として実現されとも考えられる。

本発明の多くの実施例が説明された。それにも関わらず、本発明の精神および適用範囲から逸脱することなく、さまざまな修正を加えることができることが理解されるだろう。したがって、本発明が特に説明された実施例によって制限されるのではなく、添付された請求の範囲によってだけ制限されることが理解されな

ければならない。



【図1】

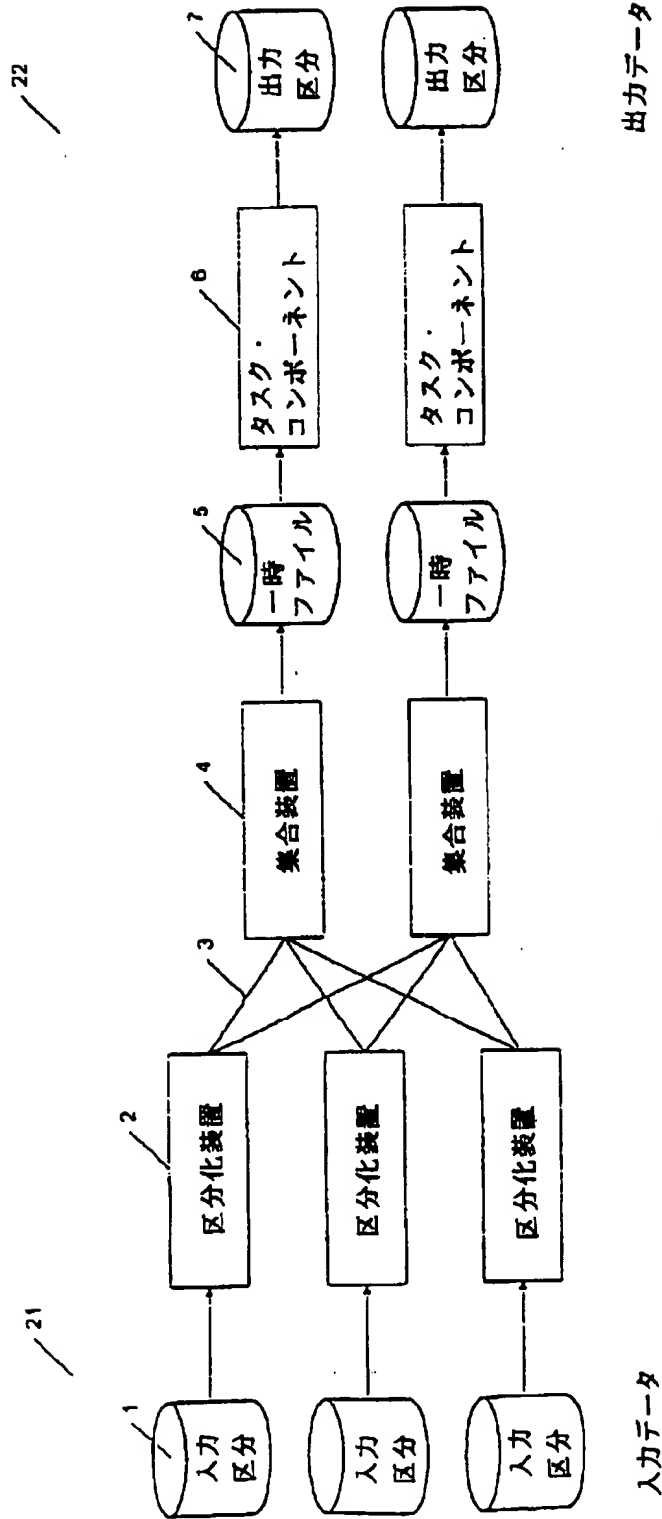


FIG. 1  
(従来の技術)

【図2】

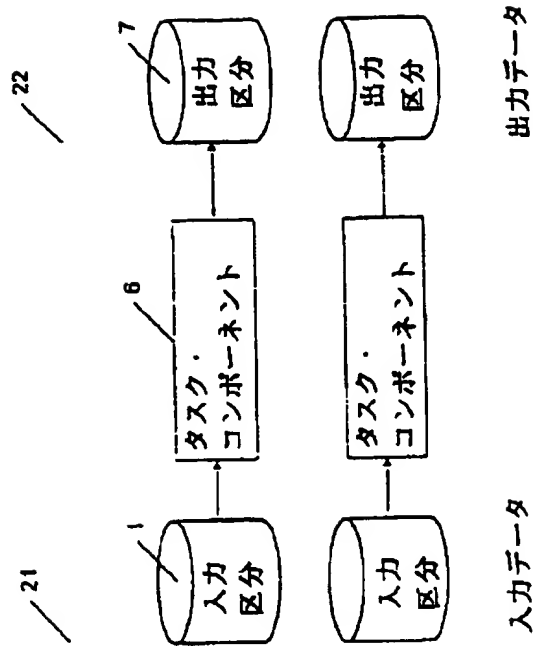


FIG. 2  
(従来の技術)

【図3】

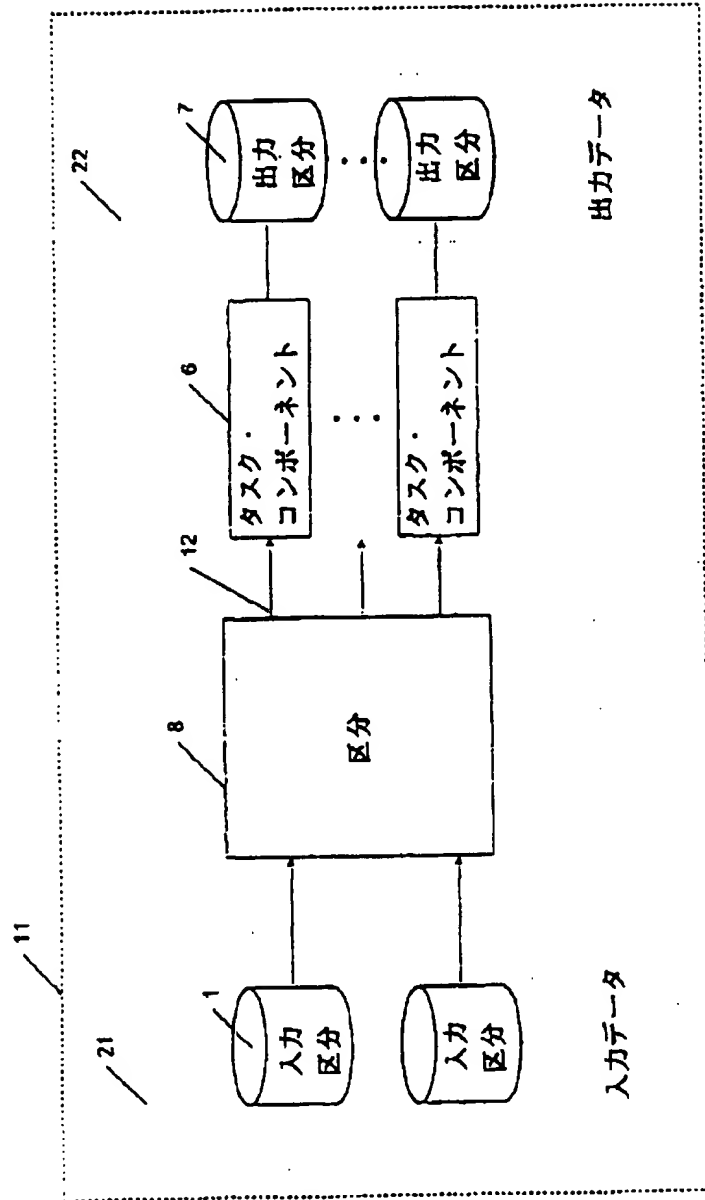


FIG. 3a  
(従来の技術)

【図 3】

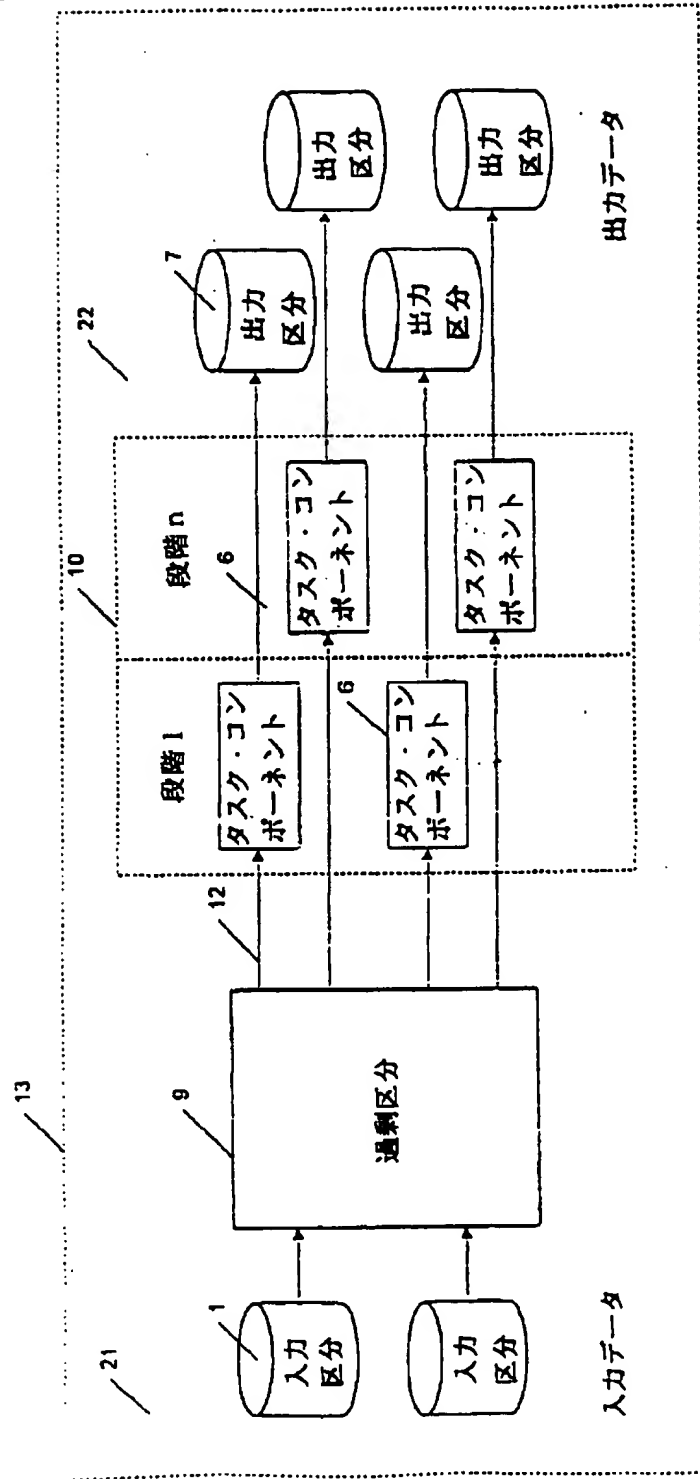


FIG. 3b

【図4】

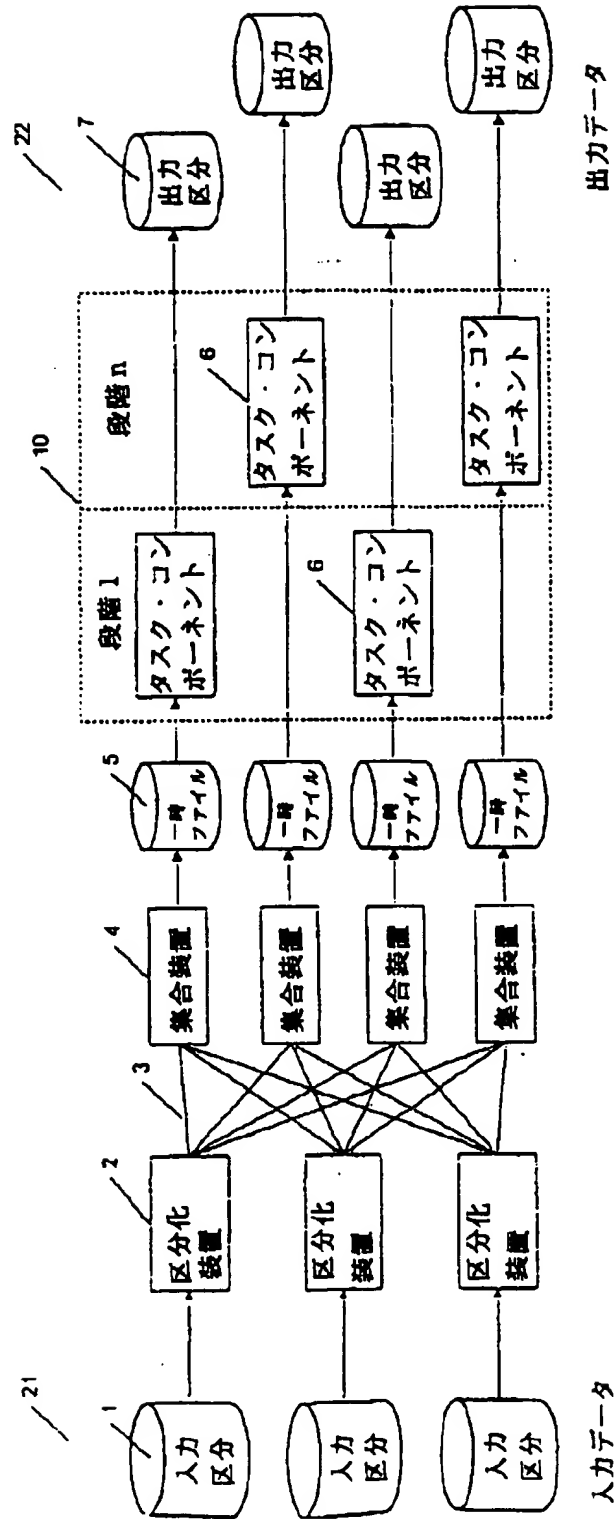


FIG. 4

【図5】

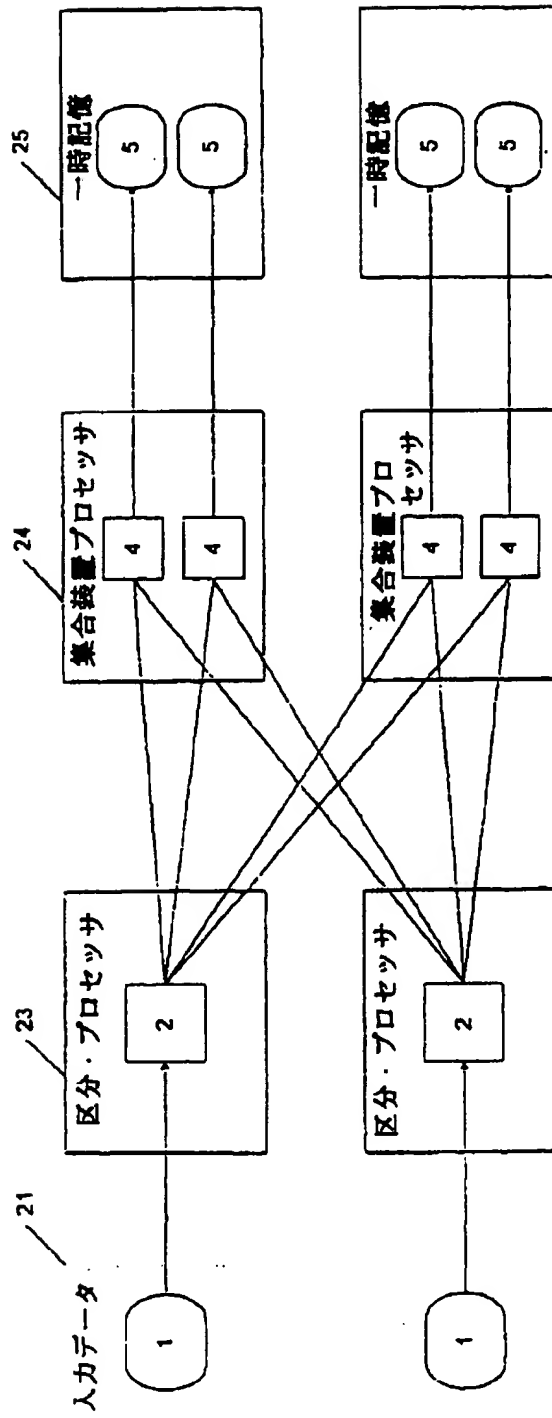


FIG. 5a

【図5】

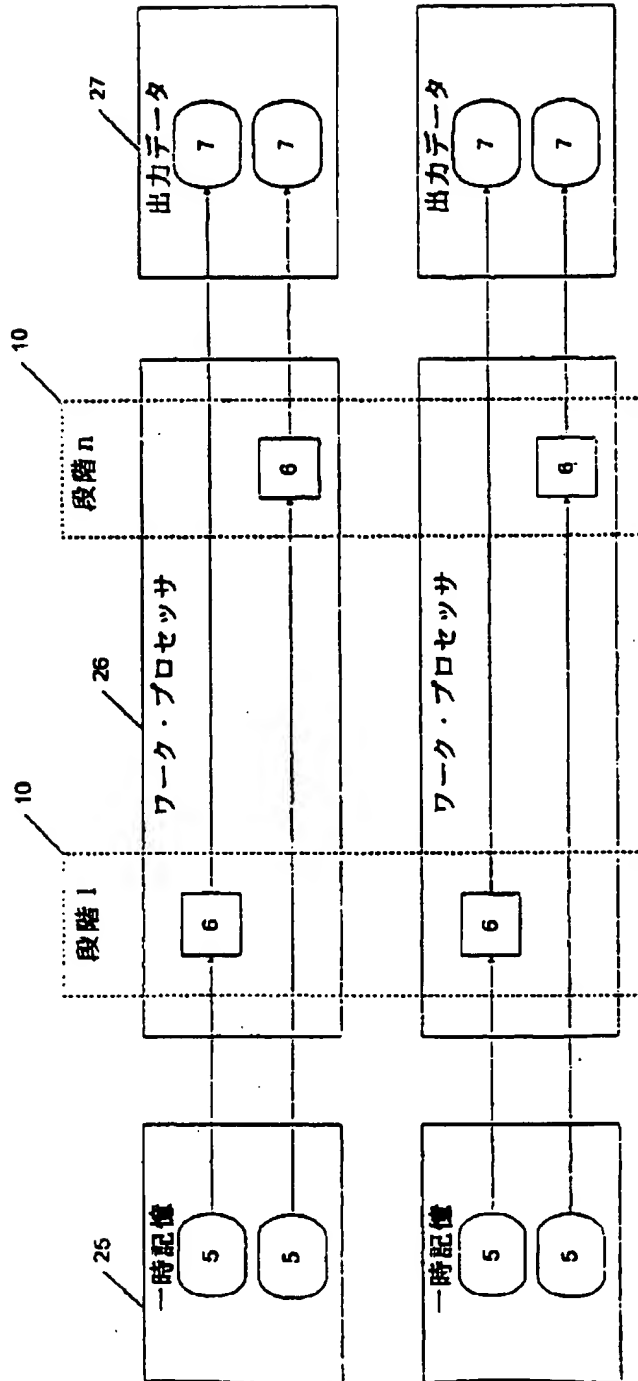


FIG. 5b

【図6】

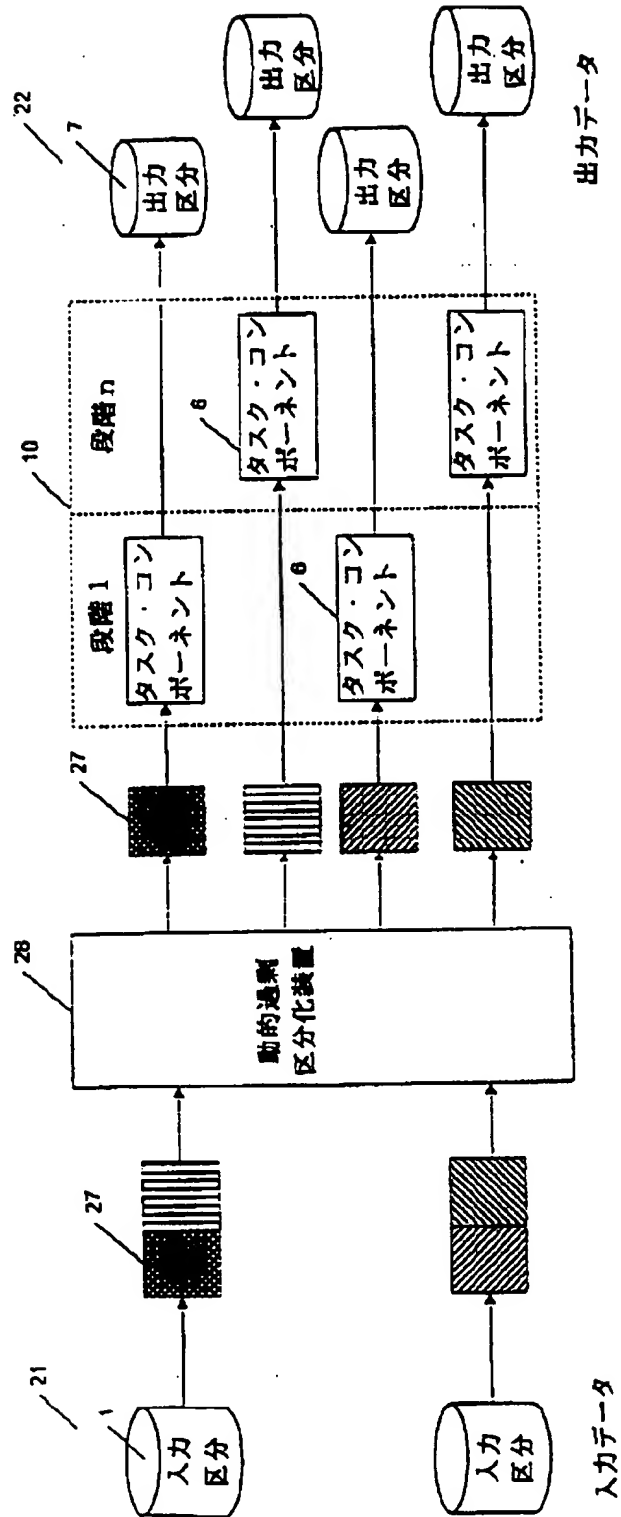


FIG. 6



【図7】

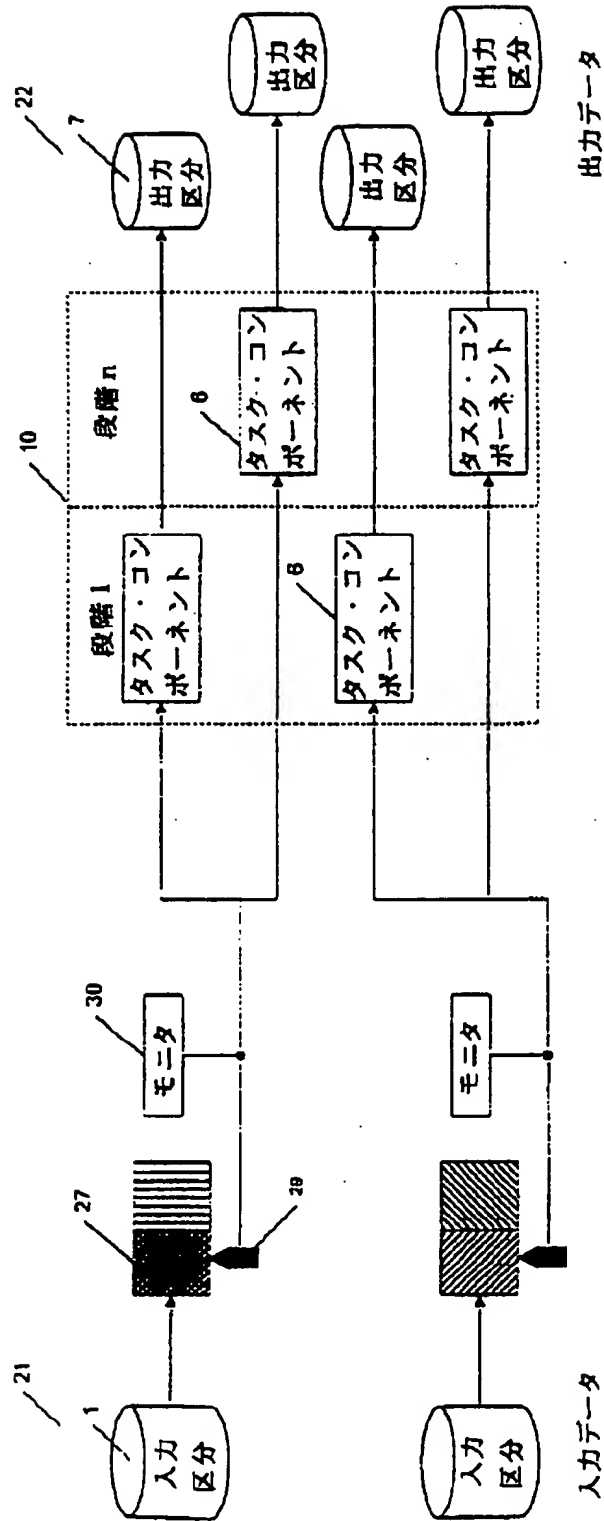
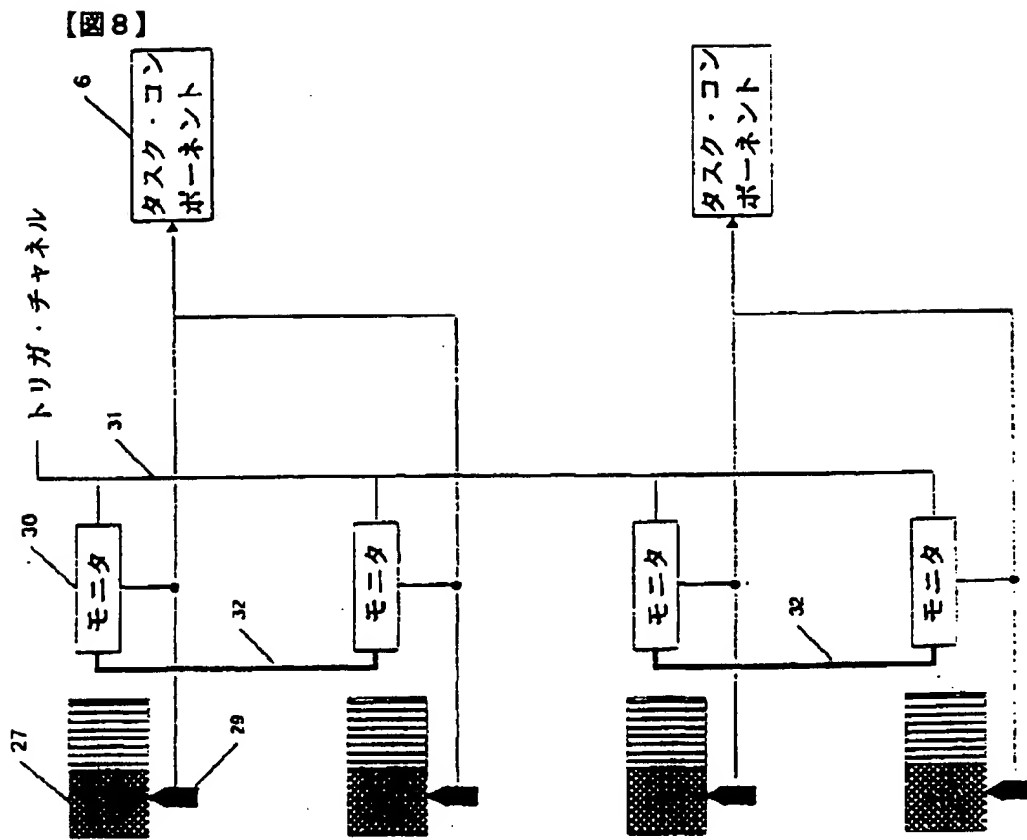


FIG. 7



**【手続補正書】****【提出日】** 1998年7月7日**【補正内容】****請求の範囲**

1. (a) なんらかの整数過剰区分化係数  $n$  に対して、 $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割することによって、 $q$  個のデータ入力を過剰区分化するステップと、

(b) コンポーネント・プログラムの各インスタンスが  $n * p$  個のデータ区分の内の 1 つを読み取り、出力データの対応する区分を作成する、直列の  $n$  個の実行段階でコンポーネント・プログラムの  $p$  個のインスタンスを実行するステップと、

(c)  $n$  個の実行段階のそれぞれの最後でチェックポイント設定を可能にするステップと、

を含む、並列処理システムで実行可能なコンピュータ・アプリケーションでチェックポイントの頻度を増加するための方法。

2.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割するステップが、さらに、

(a) データの  $n * p * q$  個のチャネルを出力するように構成される区分化プログラムを、 $q$  個のデータ入力に適用するステップと、

(b)  $n * p$  個のデータ区分を生成するために、区分化プログラムによって出力されるデータの  $n * p * q$  個のチャネルに集合プログラムを適用するステップと、

を含む、請求項 1 記載の方法。。

3.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割するステップが、さらに、

(a)  $q$  個のデータ入力を  $n$  個のデータ・セグメントに分割するために  $p$  個のカーソルを使用するステップと、

(b) 各データ・セグメントの最後を示すためにファイル終了信号を回帰的に生成するステップと、

を含む、請求項1記載の方法。。

4. 並列処理システム上で実行可能なコンピュータ・アプリケーションでのチェックポイントの頻度を増加するためのコンピュータ・プログラムであって、

(a) なんらかの整数過剰区分化係数  $n$  に対して、 $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割することによって、 $q$  個のデータ入力を過剰区分化する機能と、

(b) コンポーネント・プログラムの各インスタンスが  $n * p$  個のデータ区分の内の1つを読み取り、出力データの対応する区分を作成する、直列の  $n$  個の実行段階でコンポーネント・プログラムの  $p$  個のインスタンスを実行する機能と、

(c)  $n$  個の実行段階のそれぞれの最後でチェックポイント設定を可能にする機能と、

を実行するためにコンピュータ・システムによって読み取られ、実行されると、コンピュータ・システムを構成する、コンピュータ・システムによって読取可能な媒体に記録されるコンピュータ・プログラム。

5.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割する機能が、さらに、

(a) データの  $n * p * q$  個のチャネルを出力するように構成される区分化プログラムを、 $q$  個のデータ入力に適用する機能と、

(b)  $n * p$  個のデータ区分を生成するために、区分化プログラムによって出力されるデータの  $n * p * q$  個のチャネルに集合プログラムを適用する機能と、を含む、請求項4記載のコンピュータ・プログラム。

6.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割する機能が、さらに、

(a)  $q$  個のデータを  $n$  個のデータ・セグメントに分割するために  $p$  個のカーソルを使用する機能と、

(b) 各データ・セグメントの最後を示すためにファイル終了信号を回帰的に

生成する機能と、

を含む、請求項4記載のコンピュータ・プログラム。

7. (a) なんらかの整数過剰区分化係数  $n$  に対して、 $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割することによって  $q$  個のデータ入力を過剰区分化する機能と、

(b) コンポーネント・プログラムの各インスタンスが  $n * p$  個のデータ区分の内の 1 つを読み取って、出力データの対応する区分を作成する、直列の  $n$  個の実行段階でコンポーネント・プログラムの  $p$  個のインスタンスを実行する機能と、

(c)  $n$  個の実行段階のそれぞれの最後でチェックポイント設定を可能にする機能と、

を実行するために、記録媒体が特定の事前に定義された方法でコンピュータを動作させるように、並列処理システム上で実行可能なコンピュータ・アプリケーションでのチェックポイントの頻度を増加するためにコンピュータ・プログラムで構成されるコンピュータ読取可能な記録媒体。

8.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割する機能が、さらに、

(a) データの  $n * p * q$  個のチャネルを出力するように構成される区分化プログラムを、 $q$  個のデータ入力に適用する機能と、

(b)  $n * p$  個のデータ区分を生成するために、区分化プログラムによって出力されるデータの  $n * p * q$  個のチャネルに集合プログラムを適用する機能と、を含む、請求項 7 記載のコンピュータ読取可能な記録媒体。

9.  $q$  個のデータ入力を  $n * p$  個のデータ区分に機能的に分割する機能が、さらに、

(a)  $q$  個のデータ入力を  $n$  個のデータ・セグメントに分割するために  $p$  個のカーソルを使用する機能と、

(b) 各データ・セグメントを示すためにファイル終了信号を回帰的に生成す

る機能と、

を含む、請求項 7 記載のコンピュータ読取可能な記録媒体。

10. コンピュータ読取可能な媒体に常駐し、並列処理システム上で実行可能

なコンピュータ・アプリケーションでのチェックポイントの頻度を増加するためのコンピュータ・プログラムであって、コンピュータに、

(a) なんらかの整数過剰区分化係数 $n$ に対して、 $q$ 個のデータ入力を $n * p$ 個のデータ区分に機能的に分割することによって、 $q$ 個のデータ入力を過剰区分化させ、

(b) コンポーネント・プログラムの各インスタンスが $n * p$ 個のデータ区分の内の1つを読み取り、出力データの対応する区分を作成する、直列の $n$ 個の実行段階でコンポーネント・プログラムの $p$ 個のインスタンスを実行させ、

(c)  $n$ 個の実行段階のそれぞれの最後でチェックポイント設定を可能にさせる、

命令を備えるコンピュータ・プログラム。

11. コンピュータに $q$ 個のデータ入力を $n * p$ 個のデータ区分に機能的に分割させる命令が、さらに、コンピュータに

(a) データの $n * p * q$ 個のチャンネルを出力するように構成される区分化プログラムを、 $q$ 個のデータ入力に適用させ、

(b)  $n * p$ 個のデータ区分を生成するために、区分化プログラムによって出力されるデータの $n * p * q$ 個のチャンネルに集合プログラムを適用させる、

命令を含む請求項10記載のコンピュータ・プログラム。

12. コンピュータに $q$ 個のデータ入力を $n * p$ 個のデータ区分に機能的に分割させる命令が、さらに、コンピュータに

(a)  $q$ 個のデータを $n$ 個のデータ・セグメントに分割するために $p$ 個のカーソルを使用させ、

(b) 各データ・セグメントの最後を示すためにファイル終了信号を回帰的に

生成させる、

命令を含む請求項10記載のコンピュータ・プログラム。

## 【国際調査報告】

## INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US96/19706

## A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) : G06F 11/00, 11/08, 11/267

US CL : 395/118, 181, 673, 674

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/118, 181, 673, 674

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched  
IEEE Data base

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)  
APS, IEEE Database

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	PEERCY et al, 1995 International Symposium, 27-30 June 1995, "Software Schemes of Reconfiguration and Recovery in Distributed Memory Multicomputers Using the Actor Model"	1-2, 4-5 and 7-8
Y	US, 5,363,603 A (GLEESON) 08 November 1994, col.2, lns. 1-3, col 7-8, col. 4, lns. 41-52	1-9

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

## \* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"B" earlier document published on or after the international filing date

"I" documents which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T"

later documents published after the international filing date or priority date and not in conflict with the application but cited to understand the principles or theory underlying the invention

"X"

document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y"

document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"A"

document member of the same patent family

Date of the actual completion of the international search

06 MAY 1997

Date of mailing of the international search report

25 JUN 1997

Name and mailing address of the ISA/US  
Commissioner of Patents and Trademarks  
Box PCT  
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer  
Kevin Kriska

Telephone No. (703) 305-3804

---

フロントページの続き

(81)指定国 EP(AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OA(BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), AP(KE, LS, MW, SD, SZ, UG), UA(AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN

(72)発明者 レッサー, クリフ

アメリカ合衆国 02139 マサチューセッツ州, ケンブリッジ, リー ストリート 15, アpartment ナンバー 1

(72)発明者 ローディ, ロバート

アメリカ合衆国 01778 マサチューセッツ州, ウェイランド, バウ ロード 33

【要約の続き】

を読み取り、出力データの1つの区分を作成する。

(3) n個の実行段階のそれぞれの最後で、システムは静止しており、チェックポイント設定をすることができる。第1実施例は、過剰区分化された中間ファイルを作り出すために、既知の区分化装置プログラム、通信チャネル、および集合装置プログラムを使用して入力データを明示的に過剰区分化する。第2実施例は、オリジナル入力データの連続サブセットを連続的に読み取るためにコンポーネント・プログラムを手配することによって入力データを動的に過剰区分化する。



**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☒ **FADED TEXT OR DRAWING**
- ☒ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**